Contents

# Mutual TLS Migration

4 minute read ✓ page test

This task shows how to ensure your workloads only communicate using mutual TLS as they are migrated to Istio.

Istio automatically configures workload sidecars to use mutual TLS when calling other workloads. By default, Istio configures the destination workloads using `PERMISSIVE` mode. When `PERMISSIVE` mode is enabled, a service can accept both plain text and mutual TLS traffic. In order to only allow mutual TLS traffic, the configuration needs to be changed to `STRICT` mode.

You can use the Grafana dashboard to check which workloads are still sending plaintext traffic to the workloads in `PERMISSIVE` mode and choose to lock them down once the migration is done.

# Before you begin

- Understand Istio authentication policy and related mutual TLS authentication concepts.
- Read the authentication policy task to learn how to configure authentication policy.
- Have a Kubernetes cluster with Istio

installed, without global mutual TLS enabled (for example, use the `default` configuration profile as described in installation steps).

In this task, you can try out the migration process by creating sample workloads and modifying the policies to enforce STRICT mutual TLS between the workloads.

## Set up the cluster

- Create two namespaces, `foo` and `bar`, and deploy httpbin and sleep with sidecars on both of them:

```
$ kubectl create ns foo
$ kubectl apply -f <(istioctl kube-inject -f @samples/httpbin/httpbin.yaml@) -n foo
$ kubectl apply -f <(istioctl kube-inject -f @samples/sleep/sleep.yaml@) -n foo
$ kubectl create ns bar
$ kubectl apply -f <(istioctl kube-inject -f @samples/httpbin/httpbin.yaml@) -n bar
$ kubectl apply -f <(istioctl kube-inject -f @samples/sleep/sleep.yaml@) -n bar
```

- Create another namespace, `legacy`, and deploy `sleep` without a sidecar:

```
$ kubectl create ns legacy
$ kubectl apply -f @samples/sleep/sleep.yaml@ -n legacy
```

- Verify the setup by sending http requests (using curl) from the sleep pods, in namespaces `foo`, `bar` and `legacy`, to `httpbin.foo` and `httpbin.bar`. All requests should succeed with return code 200.

```
$ for from in "foo" "bar" "legacy"; do for to i
n "foo" "bar"; do kubectl exec "$(kubectl get p
od -l app=sleep -n ${from} -o jsonpath={.items.
.metadata.name})" -c sleep -n ${from} -- curl h
ttp://httpbin.${to}:8000/ip -s -o /dev/null -w
"sleep.${from} to httpbin.${to}: %{http_code}\n
"; done; done
sleep.foo to httpbin.foo: 200
sleep.foo to httpbin.bar: 200
sleep.bar to httpbin.foo: 200
sleep.bar to httpbin.bar: 200
sleep.legacy to httpbin.foo: 200
sleep.legacy to httpbin.bar: 200
```

If any of the curl commands fail, ensure that there are no existing authentication policies or destination rules that might interfere with requests to the httpbin service.

```
$ kubectl get peerauthentication --a
ll-namespaces
No resources found
```

```
$ kubectl get destinationrule --all-namespaces
No resources found
```

# Lock down to mutual TLS by namespace

After migrating all clients to Istio and injecting the Envoy sidecar, you can lock down workloads in the `foo` namespace to only accept mutual TLS traffic.

```
$ kubectl apply -n foo -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: "default"
spec:
  mtls:
    mode: STRICT
EOF
```

Now, you should see the request from `sleep.legacy` to `httpbin.foo` failing.

```
$ for from in "foo" "bar" "legacy"; do for to in "f
oo" "bar"; do kubectl exec "$(kubectl get pod -l ap
p=sleep -n ${from} -o jsonpath={.items..metadata.na
me})" -c sleep -n ${from} -- curl http://httpbin.${
to}:8000/ip -s -o /dev/null -w "sleep.${from} to ht
tpbin.${to}: %{http_code}\n"; done; done
sleep.foo to httpbin.foo: 200
sleep.foo to httpbin.bar: 200
sleep.bar to httpbin.foo: 200
sleep.bar to httpbin.bar: 200
sleep.legacy to httpbin.foo: 000
command terminated with exit code 56
sleep.legacy to httpbin.bar: 200
```

If you installed Istio with

`values.global.proxy.privileged=true`, you can use `tcpdump` to verify traffic is encrypted or not.

```
$ kubectl exec -nfoo "$(kubectl get pod -nfoo -lapp=httpbin -ojsonpath={.items..metadata.name})" -c istio-proxy -- sudo tcpdump dst port 80  -A
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

You will see plain text and encrypted text in the output when requests are sent from `sleep.legacy` and `sleep.foo` respectively.

If you can’t migrate all your services to Istio (i.e., inject Envoy sidecar in all of them), you will need to continue to use `PERMISSIVE` mode. However, when configured with `PERMISSIVE` mode, no authentication or authorization checks will be performed for plaintext traffic by default. We recommend you use Istio Authorization to configure different paths with different authorization policies.

## Lock down mutual TLS for the entire mesh

```
$ kubectl apply -n istio-system -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: "default"
spec:
  mtls:
    mode: STRICT
EOF
```

Now, both the `foo` and `bar` namespaces enforce mutual TLS only traffic, so you should see requests from `sleep.legacy` failing for both.

```
$ for from in "foo" "bar" "legacy"; do for to in "foo" "bar"; do kubectl exec "$(kubectl get pod -l app=sleep -n ${from} -o jsonpath={.items..metadata.name})" -c sleep -n ${from} -- curl http://httpbin.${to}:8000/ip -s -o /dev/null -w "sleep.${from} to httpbin.${to}: %{http_code}\n"; done; done
```

# Clean up the example

1. Remove the mesh-wide authentication policy.

```
$ kubectl delete peerauthentication -n istio-system default
```

1. Remove the test namespaces.

```
$ kubectl delete ns foo bar legacy
Namespaces foo bar legacy deleted.
```